Atomic Data Groups: An issue in train-test splits for the real world as demonstrated through digital hardware design

Andrew David Gunter AGUNTER@ECE.UBC.CA
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, British Columbia, Canada

Steven J.E. Wilton STEVEW@ECE.UBC.CA
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, British Columbia, Canada

Reviewed on OpenReview: *https://openreview.net/forum?id=8JfQnNEX3K*

Abstract

Machine learning (ML) has proved useful across a wide range of scientific applications. Supervised learning in particular has been successfully applied for solving prediction problems in the domain of very-large-scale integration computer-aided design (VLSI CAD), where function-based designs of digital hardware must be translated into physical designs for implementation on semiconductor devices. To avoid overestimating ML models' generalization capabilities for real-world deployments in such domains, good practices utilize realistic data and avoid test set information leakage during model preparation. In this paper we identify a further consideration in the form of atomic data groups, which are sets of very highly correlated data that may also lead to such overestimation if not accounted for in train-test splits during model evaluation. We investigate the potential impact of atomic data groups in experimental design through a case study of the VLSI CAD circuit design routing process for field-programmable gate arrays (FPGAs). Our investigations show that model performance in deployment is overestimated by 38% in this case study when atomic data groups are ignored. We hope that these results encourage other ML practitioners in different scientific domains to be critical of their train-test splits and identify when atomic data groups are relevant to their model evaluations.

Keywords: VLSI, CAD, FPGA, routing, supervised learning, information leakage

1 Introduction

In supervised learning, it is common practice to split an available dataset into a training set and test set, known as a train-test split. While the training set ideally correlates with the test set, explicit information about the test set should never enter the model training process. Most machine learning (ML) practitioners know to check against trivial errors, such as upsampling data then creating a train-test split randomly which can cause duplicate data points to appear between the train and test sets. When such errors occur, they are referred to as data leakage or information leakage from the test set. We note that the terms "data leakage" and "information leakage" see dual usage in the literature, as they can also refer to

sensitive training data being leaked through ML models in the context of data privacy and security; Del Grosso et al. (2023). In this paper, our usage of these terms is purely about leakage from test data sets into model preparation causing generalization overestimation.

Insidious forms of information leakage exist where test data isn't ever directly found in the training data, such as when model evaluations on the test data are used to make architectural or hyperparameter decisions about model changes; Samala et al. (2020). Information leakage is considered bad as it will often cause models to perform much better in closed evaluations than they do in deployment, i.e. generalization overestimation. Another potential hazard for overestimating model generalization is the usage of unrealistic data. This occurs when data gathered for a supervised learning data set does not reflect the data that a model will encounter when deployed to the real world. A model evaluated on unrealistic data cannot be trusted to generalize well. Typically, if a realistic data set is crafted and information leakage is prevented in model training and tuning then a model can be expected to generalize well in deployment. However, this is not always the case.

We identify the existence of a model generalization overestimation hazard that can occur even when using realistic data and preventing test set information leakage. This hazard occurs when a train-test split is performed without regard for what we call "atomic data groups", which are often found in the real world. We provide an explanation of what atomic data groups are and how they may occur in Section 3. We quantify the potential impact of this hazard in the context of the applied science domain of very-large-scale integration computer-aided design (VLSI CAD) for field-programmable gate arrays (FPGAs).

2 Background

Before discussing atomic data groups, we first provide information about the scientific domain which contextualizes our work.

2.1 Field-Programmable Gate Arrays

Field-programmable gate arrays (FPGAs) are semiconductor devices designed without a prescribed purpose and are thus fabricated in such a way that their functional hardware components can later be programmed when a purpose is determined. At that point, a digital circuit to execute the purpose is designed and then needs to be translated through VLSI CAD software to the FPGA's available programmable hardware components. The VLSI CAD software's output is a bitstream which is uploaded to the FPGA to program its hardware to implement the input circuit design.

2.2 VLSI CAD Routing for FPGAs

Very-large-scale integration computer-aided design (VLSI CAD) refers to a group of software tools used in the design and implementation of semiconductor devices. Implementing a digital design as a bitstream for an FPGA is a multi-stage process, with routing being the final stage. Routing determines how communicating components will be connected through the FPGA's programmable wiring. This is an NP-complete graph problem composed of many—in practice, millions of—shortest path-finding subproblems. The current state-of-the-art FPGA routing algorithm, "negotiated congestion", represents components as nodes

and wiring connectivity as edges. All communicating component (node) pairs must be assigned an exclusive path to each other while minimizing path length, but path minimization promotes illegal path sharing ("congestion").

Negotiated congestion is an iterative algorithm that progressively converges to a viable routing solution. It has three major deficiencies: (1) convergence is not guaranteed; some circuit problems are "unroutable" (2) it cannot detect when a circuit is unroutable; negotiated congestion can run infinitely without converging (3) when convergence is possible, there's no indicator for how many iterations are required. We previously researched ML tools in Gunter and Wilton (2023) that operate after each iteration to predict if negotiated congestion will converge and, if so, how many more iterations will be required. These predictions enable a doomed routing run to be intelligently exited early, preventing time being wasted on an unsolvable problem.

In the context of our work there are two notable VLSI CAD hyperparameters in FPGA routing, the VLSI CAD software's random seed and the FPGA's "channel width". The random seed influences how routing progresses. The "channels" of an FPGA are its reprogrammable wiring elements, corresponding to the edges in the routing problem graph. Increasing the utilized channel width of the device increases the number of edges in the routing graph and makes the routing problem easier.

3 Atomic Data Groups

3.1 General

An atomic data group is a collection of data points, from some larger dataset, that are all to be used exclusively in either the training set or testing set but never both. Their "atomic" nature is such that they should generally not be split; if one data point of the group is assigned to a given set then all data points will be, ignoring the trivial case of removing data points from an experiment. In the supervised learning context, each data point is an input feature object with an associated label(s). While the number of data points in an atomic data group may vary arbitrarily, no individual data point may belong to multiple atomic data groups. Although we discuss train-test splitting here, similar principles apply to train-test-validate schemes or other multi-set experiments wherein data points from an atomic data group should not be split between any two sets.

The motivation for organizing data points into atomic groups is to maintain realistic train-test splits. Typically, atomic data groups will be collected from the real world in scenarios where data points co-occur or exist together sequentially. While we cannot exhaustively describe such possibilities, examples of co-occurrence could be when multiple data points are extracted from a single physical phenomenon or the case of a data pipeline where labelling is done in batches rather than individually. The frames of a video feed which are only incorporated into a model's learning at the end of every week is an example where sequential data might be best treated as an atomic data group. The reason for this is because the freedom to assign the data points from such groups arbitrarily between the training and test sets exists only in abstract experimentation; the real world will dictate the content of the test set in deployment. In this weekly video feed example, if significant changes systematically occur mid-week then offline experiments may be prone to overestimating model generalization by splitting data points from the same week between training

and testing. In essence, accommodating for atomic data groups is beneficial in experimental design when such data are believed to be highly correlated in a manner which will not be achievable between the final training set and the data to be encountered in deployment. This paper examines the case of atomic data groups found in VLSI CAD routing for FPGAs.

3.2 FPGA Routing Data

We now use FPGA routing as an example to clarify how atomic data groups may arise. On each iteration of routing a given digital circuit design, a ML sample is extracted. Each sample consists of a feature vector paired with a regression label. The regression label for an iteration is the number of iterations remaining until the routing problem will be successfully solved. These data samples are used in Gunter and Wilton (2023) to train regression models that predict, on each iteration, how many iterations are remaining in a given FPGA routing problem for the negotiated congestion algorithm to converge on a solution. In deployment, if the models predict that convergence will require an undesirably large number of iterations (thus requiring a lot of time) then the routing problem is exited early.

The atomicity of the described data (samples) comes from the fact that they are all acquired effectively simultaneously in groups from the same routing problem attempt. It is unrealistic to expect that half of a routing problem's data is captured in a training set and the remaining half is waiting out in the real world. Correspondingly, it is unrealistic to assign half of a recorded routing problem's data to training and half to testing.

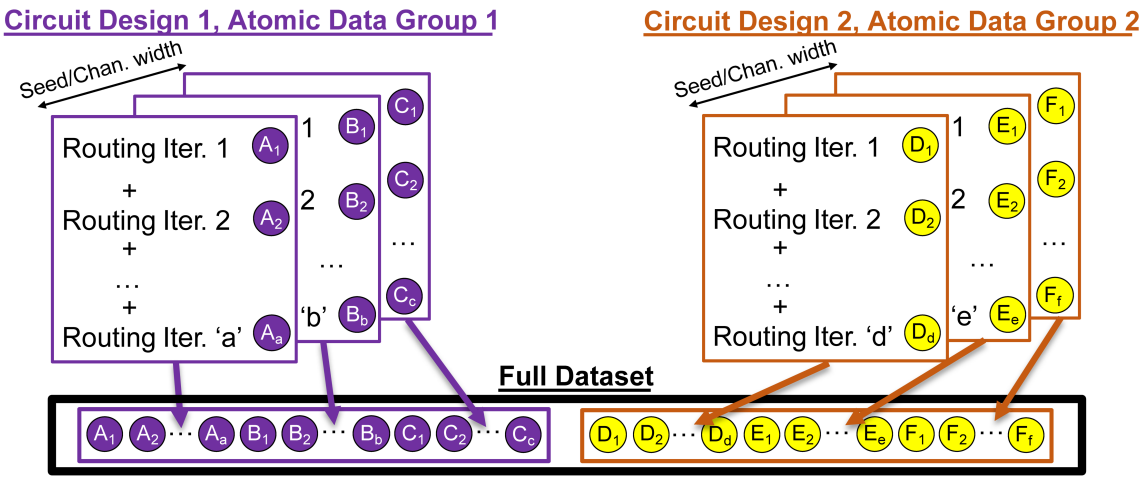

Figure 1: Atomic data groups in FPGA routing visualization

Figure 1 visualizes the atomic data groups in the FPGA routing process with atomic data groups shown in boxes and we may focus on just one. For example, purple data samples $\{A_1, A_2, ..., A_a\}$ correspond with the data extracted from one routing problem based on "Circuit Design 1". These are clearly an atomic data group by themselves.

What happens if we make variations to the VLSI CAD process by changing its random seed or the FPGA channel width then again attempt routing the same Circuit Design 1? This is represented with the data samples $\{B_1, B_2, ..., B_b\}$ and $\{C_1, C_2, ..., C_c\}$. Should these be three separate atomic data groups or are they really just three parts of one larger atomic data group? It depends on context. The motivation for identifying an atomic data group is

to effectively create train-test splits that reflect real-world deployment. If a ML model will be making predictions in deployment on a routing problem which is merely a variation of another problem available in training (i.e. from similar circuit designs) then it is appropriate to consider data from variations of routing problems as non-atomic. This corresponds with the "exceptional case" in Section 3.1, which makes assumptions that may not always hold.

4 Case Study: Data Atomicity Experiments in FPGA Routing Early Exit

Now we investigate a case study in FPGA routing early exit based on Gunter and Wilton (2023). We observe how the results of ML model evaluation vary when atomic data groups are maintained versus when they are split between training and test sets.

4.1 Experimental Setup

We extract data by routing digital circuit designs found in standard open-source academic suites. The "MCNC", "VTR", and "Titan-Other" suites contain relatively small, simple circuit designs; Yang (1991); Luu et al. (2014). The "Koios" deep learning and "Titan23" suites contain larger, more complex designs which mimic modern industrial designs; Murray et al. (2015); Arora et al. (2021). This yields a total of 126 designs. All routing is done through the VLSI CAD software tools from Murray et al. (2020) and all routing problem attempts are stopped if unsolved after 1000 iterations of negotiated congestion. Data samples are extracted as described in Section 3.2, with the same features as in Gunter and Wilton (2023), but we extract additional binary classification labels which indicate for each sample whether its corresponding routing problem was solved within 1000 routing iterations. For each of the 126 circuit designs, we vary the random seed of the VLSI CAD software 5 times and we also vary the FPGA channel width 5 times for each seed, yielding 3150 different FPGA routing problems and over 10^6 extracted data samples.

Our case study observes a regressor and a binary classifier trained on labels of the corresponding type. Both models are ensembles of gradient boosted decision trees implemented through Scikit-learn 1.0.1 with default settings; Pedregosa et al. (2011). With these models, we perform three simple experiments of 5-fold cross-validation where each experiment forms atomic data groups differently. The recipe for the experiments is that each fold is formed by determining which atomic data groups will have their entire data in the fold's test set, while the remaining data is used for training. No two samples originating from the same atomic data group are ever split between training and test sets for a fold. As per typical cross-validation, a data sample is never represented in more than one test set across folds. The three different experimental variations of this are:

1. Each data sample is an "atomic data group", i.e. each atomic data group has only a single data sample. Samples are assigned to folds' test sets randomly. Only data samples from the Koios and Titan design suites are used.

2. Data samples from the same circuit design are treated as atomic data groups. Only the circuit designs from the Koios and Titan design suites are used. Circuit designs are assigned to folds' test sets randomly.

3. Data samples from the same design suite are treated as atomic data groups. Only circuit designs from the Koios and Titan23 design suites are used in the test sets and they are assigned to folds randomly. The MCNC, VTR, and Titan-Other design suites are used for training.

Experiments 1 and 2 only use the Koios and Titan design suites because these are the most state-of-the-art. Experiment 3 takes the atomicity one level higher than what was described in Section 3.2. Circuits from the same VLSI CAD design suite are usually created with particular similarities, e.g. the Koios suite designs are all ML hardware accelerators. Arguably, it is even unrealistic to split designs from the same suite between training and test sets. We introduce the MCNC, VTR, and Titan-Other suites as Experiment 3 would have had fewer training samples otherwise. The number of training samples added in Experiment 3 is such that the total amounts per fold match Experiment 2.

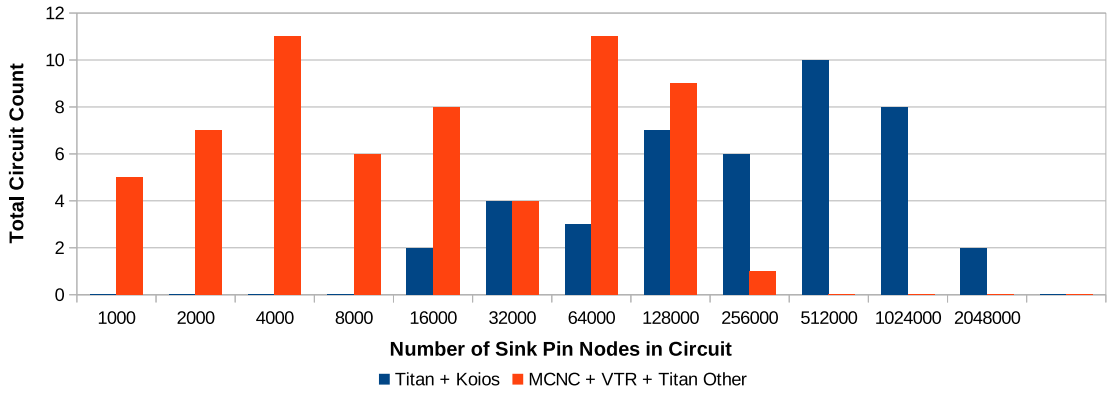

Figure 2: Comparison of circuit sizes/complexity across VLSI CAD design suites. The x-axis is histogram bin upper bound, measuring circuit size. The y-axis is the number of designs in each bin.

Figure 2 shows the differences in circuit design sizes (essentially complexity) between the design suites marked as training and testing in Experiment 3. In the real world, digital circuit designs trend toward greater complex over time and therefore it is pragmatic to prepare for a scenario where ML models for VLSI CAD will face designs of higher complexity in deployment than in training; Rapp et al. (2022).

4.2 Modelling Reality in Offline Experimental Design

Before discussing the results, it is worth clarifying the varying realism of the three experiments we have described.

It is necessarily true that Experiment 1 is unrealistic as it splits data from multiple iterations of the same routing problem attempt between the train and test sets, something which could not logically occur in deployment. There would be no logical way to train on data from a specific routing problem without first attempting to solve that routing problem and extracting its data. There is no question that Experiments 2 and 3 are more realistic in this regard, they both mimic how routing may actually proceed in real world deployment with a model trained on previous routing problem attempts' data. The primary question we wish to answer is whether the lack of realism in Experiment 1 leads to overestimation

of model generalization, i.e. whether Experiment 1 yields seemingly higher accuracy. Comparing the results of Experiment 1 with Experiment 2 will inform us of the need to consider atomic data groups in offline experimentation to replicate the logical structure of future deployment in the real world.

The secondary question to be answered is whether there is a meaningful difference in results between Experiment 2 and Experiment 3. Both experiments are modelled after plausible deployment scenarios. However, Experiment 2 assumes a deployment scenario where the circuit designs available in training are characteristically very similar to those which will be encountered in real world deployment. Experiment 3 does not make this assumption, its results could therefore be considered more general. For an experimenter who has a strong idea of the circuit designs to be encountered in deployment, Experiment 2 would appear more realistic. While Experiment 3 would appear more realistic if the experimenter were unsure. However, this is only a relevant consideration if Experiment 2 and Experiment 3 yield dissimilar results. If these results are dissimilar, then it indicates that the assumptions we are able to make about a real world deployment scenario should be considered when designing the structure of atomic data groups for offline experimentation.

4.3 Binary Classification Results

We report model accuracy but our main classification metric is the Matthews correlation coefficient (MCC) as our datasets have more negative samples (unroutable circuits) than positive ones (routable circuits) and accuracy can be misleading; Chicco and Jurman (2020).

Table 1: Effect of Data Split Organization in Classification. Higher is better.

Fold	Data	Train		Test	
Index	Split Type	Acc	MCC	Acc	MCC
0	Non-Atomic	99%	0.98	99%	0.98
0	Atomic	99%	0.98	93%	0.82
0	Suite	99%	0.94	71%	0.12
1	Non-Atomic	99%	0.98	99%	0.98
1	Atomic	99%	0.99	66%	0.44
1	Suite	99%	0.93	49%	0.23
2	Non-Atomic	99%	0.98	99%	0.98
2	Atomic	99%	0.99	83%	0.67
2	Suite	99%	0.94	58%	0.18
3	Non-Atomic	99%	0.98	99%	0.98
3	Atomic	99%	0.99	94%	0.87
3	Suite	99%	0.94	25%	0.00
4	Non-Atomic	99%	0.98	99%	0.98
4	Atomic	99%	0.99	87%	0.73
4	Suite	99%	0.94	73%	0.40
Avg	Non-Atomic	99%	0.98	99%	0.98
Avg	Atomic	99%	0.99	85%	0.71
Avg	Suite	99%	0.94	55%	0.19

Table 1 shows classification results from Experiments 1-3. Each row is a cross-validation fold result for a given data split approach. The "Fold Index" column identifies cross-

validation folds, with the "Avg" rows showing average values across all 5 folds. "Data Split Type" indicates how the data were split: "Non-atomic" for Experiment 1, "Atomic" for Experiment 2, and "Suite" for Experiment 3. "Train" and "Test" columns indicate results from evaluation on the train and test sets respectively. "Acc" is accuracy.

The non-atomic classification results in Table 1 give the first impression that the model is nearly perfect, with alarmingly constant accuracy at 99% and MCC at 0.98. The atomic data evaluation tells a different story, showing average accuracy and MCC drops of 14% and 0.28 respectively from training to testing. Atomic test results are lower because highly correlated data points have been kept together in either the training or test set. While this produces results that look inferior, it is a better reflection of reality as discussed in Section 4.2. The "Suite" results are even more negative; the atomic data group results may also overestimate model generalization. Here, average accuracy drops to 55% and the MCC is at 0.19. Fold 3 of the "Suite" data is notably bad, achieving an MCC of 0.00, equivalent to random guessing. While it is debatable which of the atomic and "Suite" results are more representative of deployment, it is certain that the non-atomic results are optimistic. Taking the atomic data as most realistic, a non-atomic evaluation overestimates the model's test MCC by 0.27 (38%). We note that our attempts at regularization by reducing model capacity actually yield worse test set results, indicating that the models are well fit to the data rather than overfit as might be inferred from these results in isolation.

4.4 Regression Results

We report the R^2 coefficient of determination for regression as it is noted to be an effective measure of model performance in Chicco et al. (2021), but we use mean absolute error (MAE) as our primary metric because it is more interpretable in our work.

Table 2: Effect of Data Organization in Regression. Lower MAE/higher R^2 is better.

Fold	Data	Train		Test	
Index	Split Type	MAE	R^2	MAE	R^2
0	Non-Atomic	56	0.79	57	0.78
0	Atomic	49	0.85	88	0.32
0	Suite	72	0.78	110	0.24
1	Non-Atomic	56	0.79	56	0.80
1	Atomic	55	0.80	77	0.62
1	Suite	72	0.79	91	0.41
2	Non-Atomic	56	0.80	56	0.79
2	Atomic	50	0.83	110	0.28
2	Suite	74	0.79	115	0.29
3	Non-Atomic	56	0.79	56	0.79
3	Atomic	57	0.80	78	-0.75
3	Suite	75	0.78	72	-0.48
4	Non-Atomic	56	0.79	56	0.79
4	Atomic	51	0.81	104	0.37
4	Suite	69	0.78	100	0.36
Avg	Non-Atomic	56	0.79	56	0.79
Avg	Atomic	52	0.82	91	0.17
Avg	Suite	72	0.78	98	0.16

Table 2 shows regression experimental results. The non-atomic results are not as strong as in Table 1 but they seemingly indicate excellent generalization from training to testing data. For this regression data, a typical label value is 500, so the MAE of 56 is fairly strong and the R^2 being around 0.79 is similarly good. A naive takeaway would be that the regression task is harder than classification but generalization is near flawless in both cases. This takeaway, of course, does not extend beyond the non-atomic results.

The atomic results in Table 2 show familiar trends, with both regression metrics degrading in the test data from the non-atomic case. The "Suite" results align with the atomic results much more closely in regression than classification but they are still even further degraded in the test data. Again taking the atomic results as most representative of deployment, the non-atomic results underestimate the MAE by 35 (38%) in the test data. It is clear that the choice of train-test split with regard to data atomicity is critical for evaluating ML model generalization here and likely also in other real-world applications.

5 Prior Work

Other works have critiqued train-test split issues before us. These works have tackled model selection problems arising from test set information leakage as we covered in Section 1 and issues in effective splitting algorithm selection for generalization estimation among others; van der Goot (2021); Birba (2020). There is further related research on data set splitting in the presence of unlabeled data which is beyond our scope; Tan et al. (2021). There are many other data split-related problems worth discussing, but our work is focused on the atomic nature of data extraction apparent in some real-world scientific applications.

While we explicitly describe atomic data groups and structured data splitting, these ideas have at least been implicit in the experimental designs of prior VLSI CAD research Xie et al. (2018); Gunter and Wilton (2023); Shrestha et al. (2022); Esmaeilzadeh et al. (2022); Tabrizi et al. (2018). Although using different language than us, more research exists in other applied science domains which identifies the possibility of model generalization overestimation hazards being contained in train-test splits; Salazar et al. (2022); Saeb et al. (2017). We are not first to identify an example of the atomic data group phenomenon, our contribution is identifying its generality and quantifying its potential impact.

6 Conclusion

In this paper, we have identified atomic data groups as a practical consideration for ML practitioners working with real-world data in a supervised learning context. We have investigated and quantified the significance of atomic data groups in the context of machine learning-driven early exit for VLSI CAD routing of FPGA digital circuit designs. For this prediction task, our experiments conducted as part of this investigation show that a failure to consider data atomicity in constructing train-test splits leads to overestimating a classifier's MCC and a regressor's mean absolute error both by 38%. To avoid overestimating ML models' generalization capabilities, researchers in other domains should be aware of the existence of atomic data groups in their own work and should make necessary accommodations in experimental design when relevant.

Broader Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Reproducibility Statement

The data required to reproduce this paper's results can be found at the corresponding Zenodo repository available through the URL: `https://zenodo.org/records/10638699?token=eyJhbGciOiJIUzUxMiIsImlhdCI6MTcwNzQ3MDc0NywiZXhwIjoxNzA5NTEwMzk5fQ.eyJpZCI6ImRhMjMwMWI5LTFhZjYtNDRkYS04MWUxLTU3NjRkOTEwYTE5YSIsImRhdGEiOnt9LCJyYW5kb20iOiI3OTAzY2NkOTY2OWZmMjA5NzEwNjU5OTMzZGJlMjVmZCJ9.ZU4V9UcCTlbSkaiIQdEB0XOqeKSTt2BjxYf252ibG6mMRTEle7h5n8aGsEy0SZg0BnIRcMDmsRU2syXZGjDMmw`. The data are broken down by classification and regression then further divided by circuit design. All machine learning work was done with Scikit-learn 1.0.1 as mentioned in Section 4.1.

References

Aman Arora, Andrew Boutros, Daniel Rauch, Aishwarya Rajen, Aatman Borda, Seyed Alireza Damghani, Samidh Mehta, Sangram Kate, Pragnesh Patel, Kenneth B. Kent, Vaughn Betz, and Lizy K. John. Koios: A deep learning benchmark suite for fpga architecture and cad research. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pages 355–362, 2021. doi: 10.1109/FPL53798.2021.00068.

Delwende Eliane Birba. A comparative study of data splitting algorithms for machine learning model selection, 2020.

Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21, 01 2020. doi: 10.1186/s12864-019-6413-7.

Davide Chicco, Matthijs Warrens, and Giuseppe Jurman. The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, 7:e623, 07 2021. doi: 10.7717/peerj-cs.623.

Ganesh Del Grosso, Georg Pichler, Catuscia Palamidessi, and Pablo Piantanida. Bounding information leakage in machine learning. *Neurocomputing*, 534:1–17, 2023. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.02.058. URL `https://www.sciencedirect.com/science/article/pii/S0925231223002072`.

Hadi Esmaeilzadeh, Soroush Ghodrati, Andrew B. Kahng, Joon Kyung Kim, Sean Kinzer, Sayak Kundu, Rohan Mahapatra, Susmita Dey Manasi, Sachin S. Sapatnekar, Zhiang Wang, and Ziqing Zeng. Physically accurate learning-based performance prediction of hardware-accelerated ml algorithms. In *2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD)*, pages 119–126, 2022. doi: 10.1109/MLCAD55463.2022.9900090.

Andrew David Gunter and Steven J.E. Wilton. A machine learning approach for predicting the difficulty of fpga routing problems. In *2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 63–74, 2023. doi: 10.1109/FCCM57271.2023.00016.

Jason Luu, Jeffrey Goeders, Michael Wainberg, Andrew Somerville, Thien Yu, Konstantin Nasartschuk, Miad Nasr, Sen Wang, Tim Liu, Nooruddin Ahmed, Kenneth B. Kent, Jason Anderson, Jonathan Rose, and Vaughn Betz. Vtr 7.0: Next generation architecture and cad system for fpgas. *ACM Trans. Reconfigurable Technol. Syst.*, 7(2), jul 2014. ISSN 1936-7406. doi: 10.1145/2617593. URL `https://doi.org/10.1145/2617593`.

Kevin Murray, Scott Whitty, Suya Liu, Jason Luu, and Vaughn Betz. Timing-driven titan: Enabling large benchmarks and exploring the gap between academic and commercial cad. *ACM Trans. Reconfigurable Technol. Syst.*, 8(2), mar 2015. ISSN 1936-7406. doi: 10.1145/2629579. URL `https://doi.org/10.1145/2629579`.

Kevin E. Murray, Oleg Petelin, Sheng Zhong, Jia Min Wang, Mohamed Eldafrawy, Jean-Philippe Legault, Eugene Sha, Aaron G. Graham, Jean Wu, Matthew J. P. Walker, Hanqing Zeng, Panagiotis Patros, Jason Luu, Kenneth B. Kent, and Vaughn Betz. Vtr 8: High-performance cad and customizable fpga architecture modelling. *ACM Trans. Reconfigurable Technol. Syst.*, 13(2), may 2020. ISSN 1936-7406. doi: 10.1145/3388617. URL https://doi.org/10.1145/3388617.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthie Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Martin Rapp, Hussam Amrouch, Yibo Lin, Bei Yu, David Z. Pan, Marilyn Wolf, and Jörg Henkel. Mlcad: A survey of research in machine learning for cad keynote paper. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41 (10):3162–3181, 2022. doi: 10.1109/TCAD.2021.3124762.

Sohrab Saeb, Luca Lonini, Arun Jayaraman, David C. Mohr, and Konrad P. Kording. The need to approximate the use-case in clinical machine learning. *GigaScience*, 6(5): gix019, 03 2017. ISSN 2047-217X. doi: 10.1093/gigascience/gix019. URL https://doi.org/10.1093/gigascience/gix019.

Jose J. Salazar, Lean Garland, Jesus Ochoa, and Michael J. Pyrcz. Fair train-test split in machine learning: Mitigating spatial autocorrelation for improved prediction accuracy. *Journal of Petroleum Science and Engineering*, 209:109885, 2022. ISSN 0920-4105. doi: https://doi.org/10.1016/j.petrol.2021.109885. URL https://www.sciencedirect.com/science/article/pii/S0920410521015023.

Ravi K. Samala, Heang-Ping Chan, Lubomir Hadjiiski, and Sathvik Koneru. Hazards of data leakage in machine learning: a study on classification of breast cancer using deep neural networks. In Horst K. Hahn and Maciej A. Mazurowski, editors, *Medical Imaging 2020: Computer-Aided Diagnosis*, volume 11314, page 1131416. International Society for Optics and Photonics, SPIE, 2020. doi: 10.1117/12.2549313. URL https://doi.org/10.1117/12.2549313.

Pratik Shrestha, Saran Phatharodom, and Ioannis Savidis. Graph representation learning for gate arrival time prediction. In *2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD)*, pages 127–133, 2022. doi: 10.1109/MLCAD55463.2022.9900101.

Aysa Fakheri Tabrizi, Logan Rakai, Nima Karimpour Darav, Ismail Bustany, Laleh Behjat, Shuchang Xu, and Andrew Kennings. A machine learning framework to identify detailed routing short violations from a placed netlist. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6, 2018. doi: 10.1109/DAC.2018.8465835.

Jimin Tan, Jianan Yang, Sai Wu, Gang Chen, and Jake Zhao. A critical look at the current train/test split in machine learning, 2021.

Rob van der Goot. We need to talk about train-dev-test splits. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4485–4494, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.368. URL https://aclanthology.org/2021.emnlp-main.368.

Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. Routenet: Routability prediction for mixed-size designs using convolutional neural network. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2018. doi: 10.1145/3240765.3240843.

Saeyang Yang. Logic synthesis and optimization benchmarks user guide version 3.0, 1991.